Optimized Time Management for Declarative Workflows

Irene Barba[1], Andreas Lanz[2], Barbara Weber[3], Manfred Reichert[2], and Carmelo Del Valle[1]

[1] Departamento de Lenguajes y Sistemas Informáticos, University of Seville, Spain
{irenebr,carmelo}@us.es
[2] Institute of Databases and Information Systems, Ulm University, Germany
{Andreas.Lanz,Manfred.Reichert}@uni-ulm.de
[3] Department of Computer Science, University of Innsbruck, Austria
Barbara.Weber@uibk.ac.at

Abstract. Declarative process models are increasingly used since they fit better with the nature of flexible process-aware information systems and the requirements of the stakeholders involved. When managing business processes, in addition, support for representing time and reasoning about it becomes crucial. Given a declarative process model, users may choose among different ways to execute it, i.e., there exist numerous possible enactment plans, each one presenting specific values for the given objective functions (e.g., overall completion time). This paper suggests a method for generating optimized enactment plans (e.g., plans minimizing overall completion time) from declarative process models with explicit temporal constraints. The latter covers a number of well-known workflow time patterns. The generated plans can be used for different purposes like providing personal schedules to users, facilitating early detection of critical situations, or predicting execution times for process activities. The proposed approach is applied to a range of test models of varying complexity. Although the optimization of process execution is a highly constrained problem, results indicate that our approach produces a satisfactory number of suitable solutions, i.e., solutions optimal in many cases.

Keywords: declarative models, temporal constraints, constraint programming, planning, scheduling, clinical guidelines

1 Introduction

Nowadays, there exists a growing interest in aligning information systems (IS) in a process-oriented way and in managing the supported processes effectively. Typically, processes are specified in an imperative way. However, declarative process models have been increasingly used allowing their users to specify *what* has to be done instead of *how* [24]. Given a declarative process model, users may choose among numerous ways to execute this model, i.e., there exist many different enactment plans for a given declarative model, each one presenting specific values for relevant objective functions (e.g., overall completion time or costs).

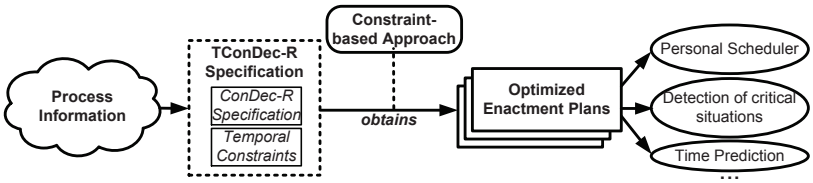

Fig. 1. Overview of our approach.

Moreover, formal specification and operational support of temporal constraints constitute fundamental challenges for any process-aware information system. In [16], we presented a set of workflow time patterns for the systematic evaluation and comparison of workflow metamodels and tools supporting temporal aspects. These time patterns are based on empirical evidence we gained from several case studies.

For supporting users working on declarative workflows with explicit temporal constraints, this paper suggests a method for generating optimized enactment plans. That is, generating plans fixing the start and end times of the activities and resources used from declarative models, while considering resources and temporal constraints. In particular, generated plans aim at optimizing given objectives (e.g., minimizing overall completion time). We built upon the work presented in [3] where we proposed an extension of the declarative language ConDec [24], named ConDec-R. This includes capabilities for reasoning about resources and parallel execution of non-preemptive activities with known duration. Moreover, we proposed an approach for generating optimized enactment plans based on ConDec-R specifications. This paper significantly extends this work by additionally supporting selected time patterns [16], i.e., temporal ConDec-R (TCondec-R) specifications are considered. Hence, higher expressiveness can be achieved and more realistic problems managed.

Figure 1 provides an overview of our approach. Taking process information as a starting point, the TConDec-R specification is defined. From this specification, optimized enactment plans can be automatically generated. For this, activities to be executed are selected and ordered (*planning problem* [14]), considering the control-flow as well as the temporal constraints imposed by the constraint-based specification. Furthermore, as stated, the generation of enactment plans related to a declarative model requires that both the temporal and the resource perspectives are considered (*scheduling problem* [7]). For planning and scheduling (P&S) the activities in a way optimizing the objective function, a constraint-based approach is used.

The generated plans can improve process support and be used for different purposes: (i) providing users with a personal schedule, allowing them to improve their performance regarding activity executions [11], (ii) facilitating early detection of critical situations through early notifications and escalations, and (iii) predicting execution times for future activities, which allows users to make informed decisions [31]. In summary, the main contributions of this paper are: (1) an extension of the approach presented in [3] (i.e., generating optimized enactment plans from ConDec-R specifications) by providing improved expressiveness through complex temporal constraints [16], and (2) the application of the proposed approach to a range of test models of varying complexity.

Section 2 introduces an application example that emphasizes the need for our approach. Section 3 gives backgrounds on related research areas. Section 4 details the

TConDec-R language and Section 5 shows how optimized plans can be generated. Section 6 deals with the evaluation, while Section 7 presents a critical discussion. Section 8 summarizes related work and Section 9 concludes the paper.

2 Application Example

To motivate the need for our approach we consider computer support for clinical guidelines. Clinical processes require the cooperation of different organizational units and medical disciplines [18]. In this context, clinical guidelines have been suggested for different medical disciplines to assist physicians in deciding about appropriate medical treatment for their patients under specific clinical circumstances [12]. Overall goal is to improve the quality of patient care and to reduce costs. Capturing respective clinical knowledge and incorporating it in clinical guidelines can potentially increase the effectiveness of patient treatment processes [18, 22]. In such an environment optimal process support becomes crucial. Traditional languages for modelling clinical computer-interpretable guidelines (CIGs) are of imperative nature [23, 30], which usually results in complex process models for which all possible treatment scenarios need to be prespecified. Moreover, imperative languages usually present limited capabilities to provide flexibility for modelling and executing clinical guidelines [22, 21, 26]. This constitutes a barrier for applying process management to healthcare since the state of patients usually cannot be predicted, and hence the exactly required treatment (or sequence of diagnostic and therapeutic procedures) is not known a-priori. To increase flexibility and to reduce complexity of clinical process models, declarative CIG models [22, 21] have been increasingly used to better fit with the nature of process-aware clinical IS and the requirements of the involved stakeholders [20].

In addition, temporal constraints play a fundamental role in the context of clinical guidelines [29, 9, 2, 8]. For example, for most therapeutic procedures, the execution of related activities has to obey temporal constraints concerning activity orders, activity durations, and the temporal time lags between activities. In turn, in other scenarios (e.g., drug administration), activities have to be repeated periodically. Moreover, there are implicit temporal constraints that can be derived from the control-flow of a process model (e.g., synchronization), or from the scheduling constraints of a CIG.

CIGs are usually modelled by hypothesizing their application in an environment providing all required resources; guidelines are developed at an abstract level without focusing on a specific execution context [18, 20]. This way, executing a CIG model requires that temporal constraints and the resource perspective are considered, i.e., reasoning about resource needs and availability is required. Moreover, given a declarative CIG model, clinical staff may choose among numerous ways to execute such model. The selection of an appropriate enactment plan, however, can be quite challenging since performance goals of the process should be considered and resource capacities be taken into account.

As stated, the proposed approach considers declarative models with explicit temporal constraints and resource reasoning, and hence, it is suitable for managing CIGs. However, our approach is not restricted to clinical environments, but can also be applied

to other domains where processes are rather flexible and where temporal constraints play an important role (e.g., automotive industry and flight planning [16]).

3 Background

To automatically generate optimized enactment plans from constraint-based specifications (cf. Section 3.1), the areas of constraint programming, planning, and scheduling (cf. Section 3.2) are combined in this work.

3.1 Constraint-based Process Models

In our proposal we use the declarative language ConDec [25, 24] as basis for the control-flow specification. We consider ConDec to be a suitable language, since it allows specifying process activities together with the constraints to be satisfied for correct process enactment and for achieving the specified goal. Moreover, ConDec allows specifying a wide set of process models in a simple and flexible way. ConDec-R extends ConDec with estimates and resources [3].

Definition 1. *A **constraint-based process model** $S = (Acts, C_{BP}, R)$ consists of a set of activities Acts, a set of constraints C_{BP}, and a set R of available resources. For each activity $a \in Acts$, resource constraints can be specified by associating the role of the required resource with that activity.*

The activities of a constraint-based process model can be executed arbitrarily often if not restricted by any constraint. ConDec templates [24] constitute parameterized graphical representations of high-level constraints between activities which can be divided into the following categories:

1. **Existence constraints**: unary relationships concerning the number of times an activity is executed. As example, *Exactly(N,A)* specifies that *A* must be executed exactly *N* times.
2. **Relation constraints**: positive binary relationships used to establish what should be executed. As example, *Precedence(A,B)* specifies that *B* may only be executed if *A* is executed beforehand.
3. **Negation constraints**: negative binary relationships used to forbid the execution of activities in specific situations. As example, *NotCoexistence(A,B)* specifies that if *B* is executed *A* cannot be executed, and vice versa.

Usually, several ways to execute constraint-based process models exist, i.e., there are different ways to execute a constraint-based process model while fulfilling all constraints. The different valid execution alternatives, however, can vary greatly in respect to their quality, i.e., in how well different performance objectives can be achieved. Thus, we propose to automatically generate optimized execution plans for a constraint-based model. We accomplish this by applying constraint programming for P&S the process activities (cf. Section 5).

3.2 Scheduling, Planning and Constraint Programming

The area of scheduling [7] includes problems for which it becomes necessary to determine an enactment plan for a set of activities related by temporal constraints. Moreover, the execution of activities requires resources, hence these activities may compete for limited resources. In general, the goal in scheduling is to find a feasible plan satisfying both temporal and resource constraints. Usually, several objective functions are considered for optimization, e.g., minimization of completion time. In a wider perspective, in AI planning [14], the activities to be executed are not established a priori, hence it becomes necessary to select them from a set of alternatives and to establish an ordering.

Constraint programming (CP) [27] has been successfully used for P&S purpose [28]. To solve a problem through CP, it needs to be modelled as a *constraint satisfaction problem* (CSP).

Definition 2. *A **CSP** $P = (V, D, C_{CSP})$ is composed out of a set of variables V, a set of domains of values D for all variables, and a set of constraints C_{CSP} between variables, such that each constraint represents a relation between a subset of variables and specifies the allowed combinations of values for these variables.*

A solution to a CSP consists of assigning values to CSP variables, such that the assignments satisfy all the constraints. Further, in CP, global constraints, i.e., constraints capturing a relation between a non-fixed number of variables, can be defined to improve the modelling of the problems.

Similar to CSPs, constraint optimization problems (COPs, cf. Def. 3) require solutions that optimize certain objective functions.

Definition 3. *A **COP** $P_o = (V, D, C_{CSP}, o)$ is a CSP including an objective function o to be optimized.*

Several mechanisms are available for solving CSPs and COPs, e.g., complete search algorithms, i.e., performing a complete exploration of a search space which is based on all possible combinations of assignments of values to the CSP variables. Regardless of the used search method, the global constraints can be implemented through filtering rules (i.e., rules responsible for removing values which do not belong to any solution) to efficiently handle the constraints in the search for solutions.

4 TConDec-R: Temporal Constraint-based Process Language

To schedule process activities when generating optimized enactment plans, ConDec-R is used (cf. Section 3.1). As motivated, we extend ConDec-R to TConDec-R (cf. Def. 4) by including templates related to selected time patterns[16]:[4] pattern *TP1* (*Time Lags between Two Activities*) enables the definition of different kinds of time lags between two activities; pattern *TP2* (*Durations*) allows specifying the duration of process elements; pattern *TP4* (*Fixed Date Element*) provides support for specifying a deadline;

[4] Since events are not specified in the considered constraint-based language, in this approach, unlike in [16], only time patterns over activities are considered.

pattern *TP5* (*Schedule Restricted Element*) allows restricting the execution of a particular element by a schedule; pattern *TP6* (*Time Based Restrictions*) allows restricting the number of times a particular process element can be executed within a predefined time frame; pattern *TP7* (*Validity Period*) allows restricting the lifetime of a process element to a given validity period; pattern *TP8* (*Time Dependent Variability*) allows varying control-flow depending on the execution time or time lags between activities/events; pattern *TP9* (*Cyclic Elements*) allows specifying cyclic elements which are performed iteratively considering time lags between cycles; and pattern *TP10* (*Periodicity*) allows specifying periodically recurring process elements according to an explicit periodicity rule (for a description of the complete set of time patterns, see [16]). Moreover, for every TConDec-R temporal template all the relations which are stated in Allen's interval algebra [1] (i.e., start-start, start-end, end-start, and end-end) can be specified.

Definition 4. *A **TConDec-R process model** $TCR = (Acts, C_T, R)$ is a constraint-based process model $S = (Acts, C_{BP}, R)$, $C_{BP} \subseteq C_T$, in which C_T includes temporal constraints.*

As example, Fig. 2(a) shows a simple TConDec-R model representing the therapy of a patient: (1) *Acts* is composed out of two activities: *A*, which has an estimated duration of 2h and requires a resource with role *R*0, and *B*, which has an estimated duration of 4h and requires a resource with role *R*1; (2) C_T is composed out of the following constraints: a) $Exactly(3, A)$, meaning that *A* must be executed exactly three times, b) $Exactly(2, B)$, expressing that *B* must be executed exactly twice, c) $DailyScheduleStart(A, [8am, 10am])$, meaning that each execution of *A* must be started between 8 am and 10 am (specific case for TP5), d) $CyclicStart - Start(B, [12h, 48h])$, meaning that between the start of two executions of *B* there must be at least 12h and at most 48h (specific case for TP9), and e) $PrecedenceEnd - Start(A, B, [2h, 4h])$, meaning that there must be a time lag of at least 2h and at most 4h between the end of any execution of *A* and the start time of the first execution of *B* (specific case for TP1); and (3) *R* is composed out of $\{[R0, 1], [R1, 1]\}$, which means that there is 1 resource with role *R*0, and 1 resource with role *R*1. In this example, all activities may be only executed between 8am and 4pm (specific case for TP5).

5 From TConDec-R to Optimized Enactment Plans

Activities and constraints are specified in a TConDec-R model. Thereby, several ways to execute this model might exist. Each of these execution alternatives leads to specific values of the objective function, i.e., the overall completion time, to be optimized. To generate optimized execution plans for a specific TConDec-R model, a constraint-based approach for P&S the process activities is proposed. This constraint-based approach includes the modelling of the declarative workflow as COP (cf. Def. 3, Section 5.1), the use of global constraints implemented through filtering rules (cf. Section 5.2), and search algorithms for solving the COP (cf. Section 5.3).

5.1 COP Model for TConDec-R Specifications

As first step, the TCondec-R model needs to be represented as CSP. Regarding the CSP model, recurring process activities (repeated activities, cf. Def. 5), which may be

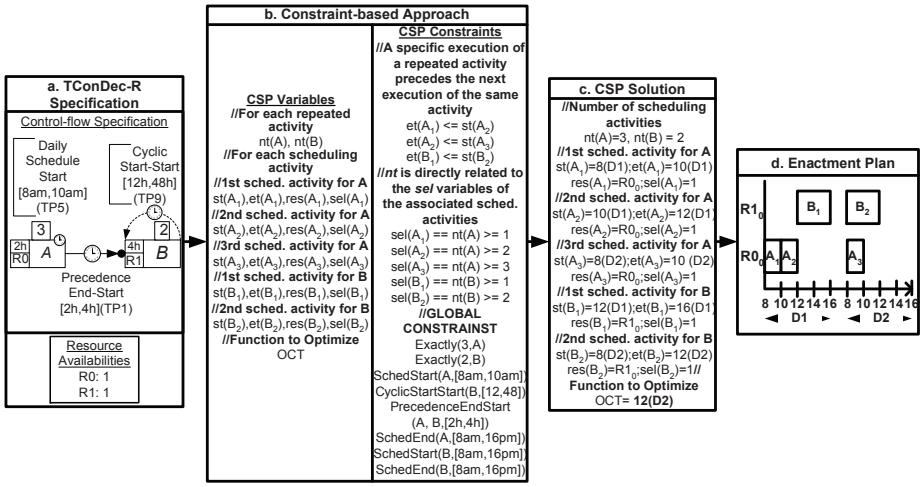

Fig. 2. From TConDec-R specification to process enactment plan.

executed arbitrarily often if not restricted by any constraint, are modelled as sequence of optional scheduling activities (cf. Def. 6). This is required since each execution of a process activity is considered as a single activity to be allocated to a specific resource and be temporarily placed in the enactment plan, i.e., stating values for its start and end times.

Definition 5. *A **repeated activity** $ra = (dur, role, nt)$ is a process activity which may be executed several times, i.e., several instances of the same activity may exist in the context of a particular process instance. A repeated activity is described by the estimated duration of the process activity (i.e., dur), the role of the required resource for activity execution (i.e., role), and a CSP variable specifying the number of times the process activity is executed (i.e., nt).*

For each repeated activity, *nt* scheduling activities exist, which are added to the CSP problem specification, apart from including a variable *nt*.

Definition 6. *A **scheduling activity** $a_i = (st, et, res, sel)$ represents the i-th execution of a repeated activity a, i.e., a specific process activity instance, where st and et are CSP variables indicating the start/end times of activity execution (each execution of a process activity needs to be temporarily placed in the enactment plan), res is a CSP variable representing the resource used for execution, and sel is a CSP variable indicating whether the activity is selected for execution.*

Moreover, an additional CSP variable representing the overall completion time (OCT), is included in the CSP model, extending the CSP to a COP (cf. Def. 7).

Definition 7. *A **COP-TConDec-R problem** related to a TConDec-R process model $TCR = (Acts, C_T, R)$ (cf. Def. 4) is a COP $P_o = (V, D, C_{CSP}, o)$ (cf. Def. 3) where:*

- *The set of variables V is composed out of all CSP variables included in the CSP model plus the CSP variable related to overall completion time (OCT), i.e., $V = \{nt(a), a \in Acts\} \cup \{st(a_i), et(a_i), res(a_i), sel(a_i), i \in [1..nt(a)], a \in Acts\} \cup OCT$.*

```
CyclicStartStart(a,[li,ls]) is added OR bounds of st(ai) for any i changed ->
  for (int i = 1; i < UB(nt(a)); i++){
    SchedulingActivity a1 = ai;
    SchedulingActivity a2 = ai+1;
    if (LB(st(a1))+(li)>LB(st(a2))) {LB(st(a2))<-LB(st(a1))+(li)} //LB(st(ai+1))>=LB(st(ai))+li
    if (UB(st(a1))>UB(st(a2))-(li)) {UB(st(a1))<-UB(st(a2))-(li)} //UB(st(ai))<=UB(st(ai+1))-li
    if (LB(st(a2))-(ls)>LB(st(a1))) {LB(st(a1))<-LB(st(a2))-(ls)} //LB(st(ai))>=LB(st(ai+1))-ls
    if (UB(st(a2))>UB(st(a1))+(ls)) {UB(st(a2))<-UB(st(a1))+(ls)} //UB(ai+1)<=UB(st(ai+1))+ls
   }
```

Fig. 3. Filtering Rule for the CyclicStartStart Template

- *The set of constraints C_{CSP} is composed out of the global constraints (implemented by the filtering rules) related to the TConDec-R constraints included in C_T together with the constraints from the proposed CSP model*[5]*, i.e.:*
  - *A specific execution of a repeated activity precedes the next execution of the same activity, i.e., $\forall i : 1 \leq i < nt(a) : et(a_i) \leq st(a_{i+1})$ for each repeated activity $a \in Acts$.*
  - *The nt variable is directly related to the sel variables of the associated scheduling activities, i.e., $\forall i : 1 \leq i \leq nt(a) : sel(a_i) = 1 \wedge \forall i > nt(a) : sel(a_i) = 0$ for each repeated activity $a \in Acts$.*
  - *$OCT = max_{a \in Acts}(et(a_{nt(a)}))$.*
- *The set of domains D is composed out of the domains for each variable from V.*
- *The objective function to be optimized is overall completion time, i.e., $o = OCT$.*

In this way, the COP model which was proposed for ConDec-R specifications [3] has been extended by including: (1) a new global constraint for each of Allen's interval algebra relation of each specific case of every supported temporal constraint, i.e., time patterns TP2, TP4, TP5, TP6, TP7, TP8, TP9, and TP10, and (2) a new global constraint for each of Allen's interval algebra relation of every relation and negation ConDec constraint for allowing the specification of time lags (i.e., time pattern TP1). Moreover, when all process activities may be executed in a specific time frame $[li, ls]$, the constraints $DailyScheduleStart(a, [li, ls])$ and $DailyScheduleEnd(a, [li, ls])$ are included for every activity $a \in Acts$ which is not involved in any other schedule constraint (cf. Fig. 2(b)). This is needed since the *st* and *et* variables can take any value, e.g., a value corresponding to 4 am, if not restricted by any constraint.

Figure 2 also shows the translation from a TConDec-R specification into a CSP so that the CSP variables and constraints are stated as explained in Def. 7 (cf. Fig. 2(b)).

5.2 Filtering Rules

For each TConDec-R template our constraint-based proposal includes a related global constraint implemented through a filtering rule. Since we extend ConDec-R[6] [3] by time patterns, new filtering rules related to these time patterns have been developed, i.e., one filtering rule for each new global constraint (cf. Section 5.1). As examples, Fig. 3 and 4

[5] Resources are implicitly constrained since the solver which is used provides a high-level constraint modelling specific to scheduling which includes the management of shared resources.

[6] A detailed description of the ConDec-R filtering rules can be found at http://regula.lsi.us.es/MOPlanner/FilteringRules.pdf.

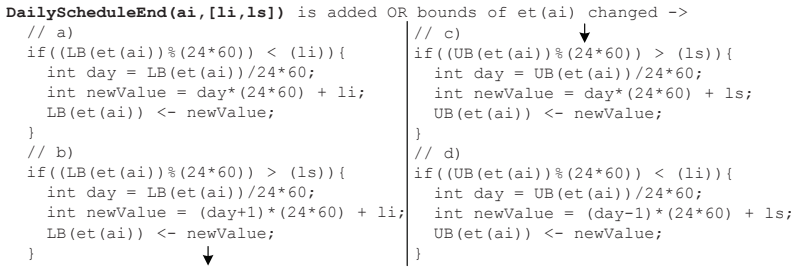

Fig. 4. Filtering Rule for the DailyScheduleEnd Template

show the filtering rules related to the $CyclicStartStart(a,[li,ls])$ and $DailySchedule-End(a,[li,ls])$[7] global constraints, where $UB(var)$ and $LB(var)$ represent the upper and lower bounds of the domain of *var*, respectively. Most of the newly developed filtering rules present a propagation reasoning similar to the one included in the ConDec-R filtering rules, i.e., they basically differ in the consideration of the time lags (see Fig. 3 for an example). However, for the filtering rules related to the schedule templates, it becomes necessary to reason about the day in which the upper and lower bounds of the start and/or end time variables are placed. Specifically, for the filtering rule of Fig. 4, for every activity execution a_i the next reasoning is carried out:[8] a) if the lower bound of $et(a_i)$ corresponds to a time of a day *d* which is lower than the time *li*, then that lower bound is updated to the time *li* of the day *d*; b) if the lower bound of $et(a_i)$ corresponds to a time of a day *d* which is greater than the time *ls*, then that lower bound is updated to the time *li* of the day after *d*; c) if the upper bound of $et(a_i)$ corresponds to a time of a day *d* which is greater than the time *ls*, then that upper bound is updated to the time *ls* of the day *d*; and d) if the upper bound of $et(a_i)$ corresponds to a time of a day *d* which is lower than the time *li*, then that upper bound is updated to the time *ls* of the day before *d*.

In this way, the constraints stated in the TConDec-R specification (cf. Def. 4) can be easily included in the CSP model through the related global constraints. Moreover, the related filtering rules increase the efficiency in the search for solutions, since during the search process these filtering rules remove inconsistent values from the domains of the variables. In the CSP model, initial estimates are made for upper and lower bounds of variable domains, and these values are refined during the search process.

5.3 Search Algorithms

Once the problem is modelled, several constraint-based mechanisms can be used to obtain the solutions of the COP (cf. Def. 3), i.e., optimized enactment plans (cf. Def. 8). For the empirical evaluation of this paper, we use the heuristic complete search method

[7] Note that since the $DailyScheduleEnd(a,[li,ls])$ constraint individually affects each activity execution, the filtering mechanism for every scheduling activity is carried out in a separated way to increase the efficiency. In this way, the $DailyScheduleEnd(a,[li,ls])$ constraint is implemented through the set $\{DailyScheduleEnd(a_i,[li,ls]), i \in [1..nt(a)]\}$ of filtering rules.

[8] To deal with different time granularities, all the temporal specifications of the TConDec-R model are automatically converted to minutes when generating the CSP.

setTimes [17] since it has demonstrated its ability to obtain good solutions to complex scheduling problems.

Definition 8. *An **enactment plan** consists of: (1) the number of times each activity is executed, (2) the start and end times for each activity execution, and (3) the resource which is used for each activity execution.*

Figure 2(d) shows an enactment plan which represents the CSP solution of Fig. 2(c) related to the TConDec-R specification of Fig. 2(a).

Since the generation of optimal plans has NP-complexity [13], it is not possible to ensure the optimality of the generated plans for all cases. The developed constraint-based approach, however, allows solving the considered problems in an efficient way, reaching solutions which are optimal in many cases (cf. Section 6).

6 Empirical Evaluation

To evaluate the effectiveness of our approach, a controlled experiment has been conducted. Section 6.1 describes the design underlying the experiment, and Section 6.2 shows the experimental results and the data analysis.

6.1 Experimental Design

Purpose: The purpose of the empirical evaluation is to analyze the behavior of our proposal in the generation of optimal enactment plans from TConDec-R (i.e., temporal ConDec-R) specifications.

Objects: Considering the application scenario from Section 2, the empirical evaluation is based on the generic TConDec-R model (cf. Fig. 5), which represents a specific treatment to be applied to #*P* patients. This scenario has been selected, since it includes typical relations present in actual CIGs. It further contains a representative set of time patterns. In this context, we presume that all activities may be executed between 8am to 8pm.

The generic TConDec-R model of Fig. 5 is specified by replacing the variables $\{\#P, \#R0, \#R1, D_{a \in Acts}\}$ with specific values, being D_a the estimated duration for a. Regarding the number of patients values $\#P \in \{5, 10, 15\}$, and for the number of resources with roles $R0$ and $R1$ values $\{5, 10\}$ are considered. In addition, different games of durations for each process activity are assumed (G), since this aspect has great influence on the complexity of the search for optimums. Note that the considered problems are an extension of typical scheduling problems. 30 instances are randomly generated for each TConDec-R model by varying activity durations between 5 and 30 minutes.[9]

Independent Variables: For the empirical evaluation, (1) the number of patients (i.e., #*P*), (2) the number of available resources with role $R0$ or $R1$, respectively (i.e., #$R0$, #$R1$), and (3) the game which establishes the activity durations (i.e., G) are taken as independent variables.

[9] The set of games which are used for the empirical evaluation are available at http://www.lsi.us.es/ quivir/irene/Games.rar.

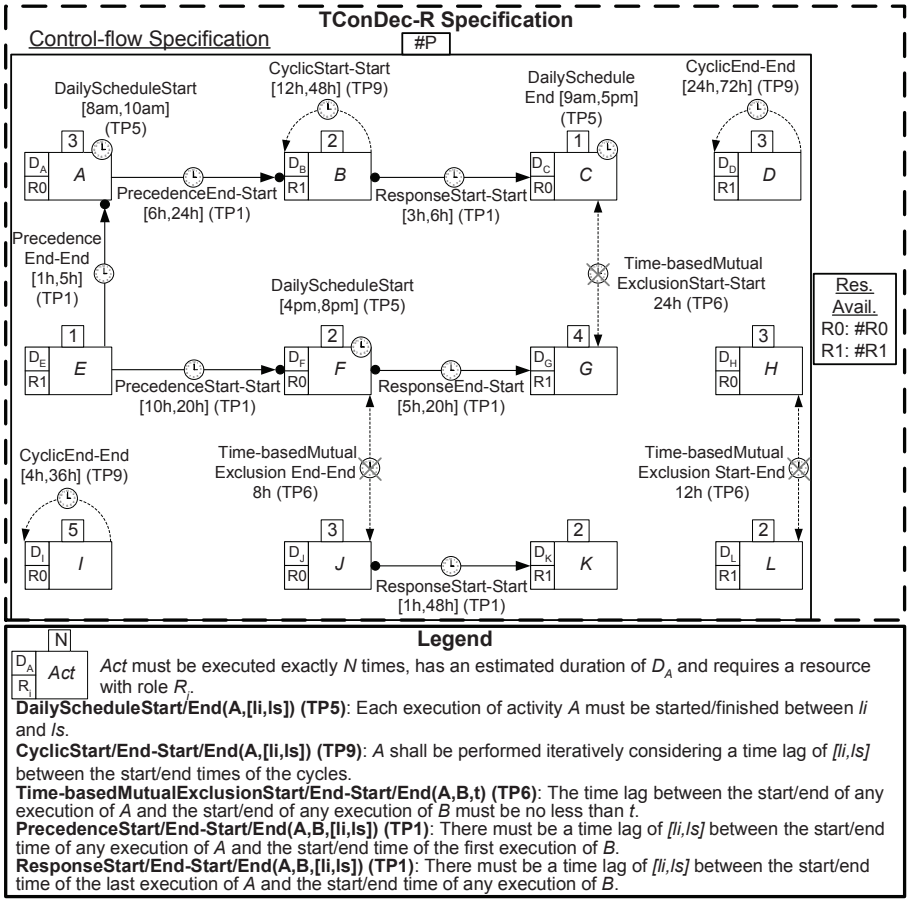

Fig. 5. A generic TConDec-R model.

Response Variables: The suitability of our approach is tested regarding the following variables: (1) percentage of optimal solutions found (i.e., $\%Opt$)[10], (2) average time (in seconds) for getting optimal solutions, considering the cases in which the optimal solution is found (i.e., $T_{Opt}(s)$), and (3) average value of the objective function obtained (i.e, overall completion time $OCT(min)$).

Experimental Design: For the model depicted in Fig. 5, 360 instances (i.e., $3 \times 2 \times 2 \times 30$) are generated considering different values of #*P* (3 values), #*R*0 (2 values), #*R*1 (2 values), and *G* (30 problem instances). The response variables are then calculated by considering the average values for the 30 problem instances.

Experimental Execution: The machine we use is an Intel Core2, 2.13 GHz, 1.97 GB memory, running on Windows XP. For the experiments, the complete search method *setTimes* [17] is run until a 5-minutes CPU time limit is reached. To implement the

[10] The optimality of the solutions can be only ensured if the search algorithm stops before reaching the time limit. Otherwise the optimality of the reached solution is unknown.

constraint-based problems (cf. Section 5), COMET [10] is used, which is able to generate high-quality solutions for highly constrained problems in an efficient way. This system provides a scheduling module offering high-level constraint modelling and search abstraction, both specific to scheduling.

6.2 Experimental Results and Data Analysis

For each problem (i.e., $\{\#P, \#R0, \#R1\}$) Table 1 shows: (1) the total number of scheduling activities (cf. Def. 6) to be planned and scheduled (*#SchedAct*) , and (2) the values of the response variables (i.e., $\%Opt$, T_{Opt}, and OCT) for the 30 problem instances randomly generated.[11] As expected, the percentage of optimal solutions found decreases and the average time for getting optimal solutions increases as the number of patients (and hence the number of scheduling activities) increases. Specifically, for 5 patients (155 scheduling activities) the optimum is found in almost all cases (the average value for $\%Opt$ is equal to 99.16%), for 10 patients (310 scheduling activities) the average value for $\%Opt$ is equal to 36.66%, and for 15 patients (465 scheduling activities) the average value for $\%Opt$ is equal to 6.66%. Moreover, in almost all cases, the value for $\%Opt$ increases and the value for $T_{Opt}(s)$ decreases as the number of available resources increases. As expected, the average value for OCT increases as the number of patients (and hence the number of scheduling activities) increases and the number of available resources decreases. Additionally, it can be seen that the number of available resources with role $R1$ seems to be more influential than the number of available resources with role $R0$ in all response variables.

In general, experimental results show that despite NP-complexity of the problems considered, the values for the percentage of optimal solutions found and for the average time for getting optimums are quite good for medium-sized problems (between 155 and 465 scheduling activities). Note that getting the optimum for scheduling problems of 155-465 activities can entail a great complexity. In fact, there are many scheduling benchmarks of smaller size for which their optimal values are not even known.

7 Discussion and Limitations

The current approach allows modelling processes in an easy way, since the considered declarative specifications are based on high-level constraints. Furthermore, time patterns can be easily specified since the proposed constraint-based language includes temporal constraints. This is a big advantage. Although temporal constraints play an important role in the context of long-running processes, time support is very limited in existing process management systems [16]. With our extension, an increased expressiveness to the specification language is provided (compared to [3]), and hence more realistic problems can be managed, e.g., CIG support in the clinical domain (cf. Section 2). Moreover, one advantage of our proposal is that optimized enactment plans are generated by considering all process activities; hence, it allows for a global optimization

[11] The set of optimized enactment plans which were generated during the empirical evaluation are available at http://www.lsi.us.es/ quivir/irene/OptimizedEnactmentPlans.rar.

Table 1. Experimental results (5-minutes time limit).

#P	*#R0*	*#R1*	*#SchedAct*	*%Opt*	$T_{Opt}(s)$	$OCT(min)$
5	5	5	155	96.66	0.21	3666
5	5	10	155	100	3.03	3618
5	10	5	155	100	0.94	3618
5	10	10	155	100	0.98	3618
10	5	5	310	3.33	0.86	4602.58
10	5	10	310	46.66	31.45	3833.66
10	10	5	310	10	0.83	4511.85
10	10	10	310	86.66	3.36	3715.33
15	5	5	465	0	-	6437.20
15	5	10	465	16.66	9.27	4590.43
15	10	5	465	3.33	1.45	6388.27
15	10	10	465	6.66	1.50	4317.93

of the objective functions. Finally, the automatic generation of optimized plans can deal with complex problems in a simple way, as demonstrated in Section 6. Hence, a wide study of several aspects can be carried out by simulation.

Nonetheless, the proposed approach also presents a few limitations. First, the analysts must deal with a new language for the constraint-based specification, thus a period of training is required to let them become familiar with TConDec-R specifications. Secondly, the optimized process models are generated by considering estimated values for the number of process instances, activity durations, and resource availability, and hence the current proposal is only appropriate for processes in which these values can be estimated. However, P&S techniques can be applied to replan the activities in the enactment phase by considering the actual values of the parameters, as stated in [4].

8 Related Work

This paper extends the approach presented in [3] by providing improved expressiveness through temporal constraints [16]. We are not aware of any other approach for generating enactment plans from declarative specifications, however, there exist some further proposals which could be extended in such direction [21, 20]. Similar to our work, [21] presents a declarative language based on ConDec (i.e., CIGDec) for the modelling and enactment of CIGs. From CIGDec specifications an automaton representing all feasible traces can be generated. The overall completion time of all the traces could be calculated [31], and hence optimized enactment plans be generated. However, as a disadvantage of this approach, generating the automaton is NP complete, and, unlike the proposed approach, no heuristics is used. Additionally, CLIMB [20] could be used to generate quality traces from declarative specifications, and calculate its completion time. Then, the best traces could be selected. Unlike our approach, [20] neither considers optimality nor resource availability. Finally, the time patterns presented in [16] are not considered in [21, 20].

Many constraint-based approaches for modelling and solving P&S problems have been proposed [27]. Moreover, several proposals exist for filtering algorithms related

to specialized scheduling constraints [5]. Therefore, the considered problem could be managed by adapting existing constraint-based approaches. However, these problems include many non-typical scheduling constraints from ConDec, which entail complex reasoning about several combined innovative aspects, such as the alternating executions of repeated activities together with the varying number of times which these activities are executed. Therefore, we implemented our own specific filtering rules to increase the efficiency in the search for solutions.

Furthermore, constraint-based approaches for process design verification have been proposed in process-aware IS [19]. Unlike our approach, they do not consider the generation of optimized process enactment plans.

Related to the clinical domain, the CIG languages presented in [21, 20, 6, 15] do not consider time patterns. However, there are approaches focussing on the treatment of temporal aspects in CIGs (e.g., [29, 9, 2, 8]). Opposed to our work, the works presented in [29, 9, 2, 8] do not consider optimality issues when managing temporal constraints.

9 Conclusions and Future Work

This paper presents a method for generating optimized enactment plans (e.g., minimizing overall completion time) from declarative temporal process models. The generated plans can be used for different purposes, e.g., providing users with a personal schedule, facilitating early detection of critical situations, or predicting execution times for process activities. The proposed approach is applied to a range of test models of varying complexity. Results indicate that, despite the NP-complexity of the considered problems, our approach produces solutions being optimal in many cases. As for future work, we will explore various constraint-based solving techniques and analyze their suitability for the generation of optimized enactment plans.

References

1. J.F. Allen. Maintaining knowledge about temporal intervals. In *Proc. Communications of the ACM*, pages 832–843, 1983.
2. L. Anselma, P. Terenziani, S. Montani, and A. Bottrighi. Towards a comprehensive treatment of repetitions, periodicity and temporal constraints in clinical guidelines. *Artificial Intelligence In Medicine*, 38:171 – 195, 2006.
3. I. Barba and C. Del Valle. A Constraint-based Approach for Planning and Scheduling Repeated Activities. In *Proc. COPLAS*, pages 55–62, 2011.
4. I. Barba, B. Weber, and C. Del Valle. Supporting the Optimized Execution of Business Processes through Recommendations. In *Proc. BPM Workshops*, pages 135–140, 2011.
5. R. Barták and O. Cepek. Incremental propagation rules for a precedence graph with optional activities and time windows. *Transactions of the Institute of Measurement and Control*, 32(1):73–96, 2010.
6. A. Bottrighi, F. Chesani, P. Mello, G. Molino, M. Montali, S. Montani, S. Storari, P. Terenziani, and M. Torchio. A hybrid approach to clinical guideline and to basic medical knowledge conformance. In *Proc. AIME*, pages 91–95, 2009.
7. P. Brucker and S. Knust. *Complex Scheduling (GOR-Publications)*. Springer, 2006.

8. C. Combi, M. Gozzi, J.M. Juarez, and B. Oliboni. Conceptual Modeling of Temporal Clinical Workflows. In *Proc. TIME*, pages 70 – 81, 2007.
9. G. Duftschmid, S. Miksch, and W. Gall. Verification of temporal scheduling constraints in clinical practice guidelines. *Artificial Intelligence In Medicine*, 25(2):93 – 121, 2002.
10. Dynadec. Comet Downloads. `http://dynadec.com/support/downloads/`, 2010. [Online; accessed 3-October-2011].
11. J. Eder, H. Pichler, W. Gruber, and M. Ninaus. Personal schedules for workflow systems. In *Proc. BPM*, pages 216–231, 2003.
12. M.J. Field and K.N. Lohr. *Guidelines for clinical practice: from development to use*. Institute of Medicine, Washington, D.C: National Academy Press, 1992.
13. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1979.
14. M. Ghallab, D. Nau, and P. Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, Amsterdam, 2004.
15. M.A. Grando, W.M.P. van der Aalst, and R.S. Mans. Reusing a Declarative Specification to Check the Conformance of Different CIGs. In *Proc. BPM Workshops*, pages 188–199, 2011.
16. A. Lanz, B. Weber, and M. Reichert. Workflow Time Patterns for Process-Aware Information Systems. In *Proc. BPMDS and EMMSAD*, pages 94 – 107, 2010.
17. C. Le Pape, P. Couronne, D. Vergamini, and V. Gosselin. Time-versus-capacity compromises in project scheduling. In *Proc. PlanSIG*, pages 498–502, 1994.
18. R. Lenz and M. Reichert. IT support for healthcare processes premises, challenges, perspectives. *Data & Knowledge Engineering*, 61(1):39–58, 2007.
19. Ruopeng Lu, Shazia Wasim Sadiq, Guido Governatori, and Xiaoping Yang. Defining adaptation constraints for business process variants. In *BIS*, pages 145–156, 2009.
20. M. Montali. *Specification and Verification of Declarative Open Interaction Models: a Logic-Based Approach*. PhD thesis, Department of Electronics, Computer Science and Telecommunications Engineering. University of Bologna, 2009.
21. N. Mulyar, M. Pesic, W.M.P. van der Aalst, and M. Peleg. Declarative and Procedural Approaches for Modelling Clinical Guidelines: Addressing Flexibility Issues. In *Proc. BPM 2007 Workshops*, pages 335–346, 2008.
22. N. Mulyar, W. M. P. van der Aalst, and M. Peleg. A pattern-based analysis of clinical computer-interpretable guideline modelling languages. *Journal of the American Medical Informatics Association*, 14:781 – 787, 2007.
23. M. Peleg and et al. Comparing computer-interpretable guideline models: A case-study approach. *Journal of the American Medical Informatics Association*, 10(1):52 – 68, 2003.
24. M. Pesic. *Constraint-Based Workflow Management Systems: Shifting Control to Users*. PhD thesis, Technische Universiteit Eindhoven, Eindhoven, 2008.
25. M. Pesic, M.H. Schonenberg, N. Sidorova, and W.M.P. van der Aalst. Constraint-Based Workflow Models: Change Made Easy. In *OTM Conferences (1)*, pages 77–94, 2007.
26. M. Reichert. What BPM Technology Can Do for Healthcare Process Support . In *Proc. AIME*, pages 2–13, 2011.
27. F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming* . Elsevier, 2006.
28. M.A. Salido. Introduction to planning, scheduling and constraint satisfaction. *Journal of Intelligent Manufacturing*, 21(1):1–4, 2010.
29. Y. Shahar. A framework for knowledge-based temporal abstraction. *Artificial Intelligence*, 90(1/2):79 – 133, 1997.
30. A. ten Teije, S. Miksch, and P. Lucas. *Computer-based Medical Guidelines and Protocols: A Primer and Current Trends*. IOS Press, 2008.
31. W.M.P. van der Aalst, M.H. Schonenberg, and M. Song. Time prediction based on process mining. *Information Systems*, 36(2):450–475, 2011.